13

Supreme Court, U.S.
FILED
DEC 7 1995
CLERK

No. 94-2003

In the Supreme Court of the United States

OCTOBER TERM, 1995

LOTUS DEVELOPMENT CORPORATION,
PETITIONER,

v.

BORLAND INTERNATIONAL, INC.,
RESPONDENT.

ON WRIT OF CERTIORARI TO THE UNITED STATES COURT OF APPEALS FOR THE FIRST CIRCUIT

BRIEF AMICUS CURIAE OF COMPUTER SCIENTISTS IN SUPPORT OF RESPONDENT

RON KILGARD
(Counsel of Record)
KARL M. TILLEMAN

Dalton, Gotto, Samson & Kilgard, P.L.C.
Suite 900
3101 North Central
Phoenix, AZ 85012
Phone: (602) 248-0088

Attorneys for Amici

December 1995

41pp

TABLE OF CONTENTS

TABLE OF AUTHORITIES

THIS PAGE INTENTIONALLY LEFT BLANK

No. 94-2003

In the Supreme Court of the United States

OCTOBER TERM, 1995

LOTUS DEVELOPMENT CORPORATION,
PETITIONER,

v.

BORLAND INTERNATIONAL, INC.,
RESPONDENT.

ON WRIT OF CERTIORARI TO THE UNITED STATES COURT OF APPEALS FOR THE FIRST CIRCUIT

BRIEF AMICUS CURIAE OF COMPUTER SCIENTISTS IN SUPPORT OF RESPONDENT

I. INTRODUCTION AND INTEREST OF AMICI

This amicus brief is filed on behalf of numerous individual computer scientists who believe the opinion of the U.S. Court of Appeals for the First Circuit was correctly decided and should be upheld. The signatories to this brief include some of the leading, pioneering scientists in the computer industry. We have invented or contributed to the authorship of numerous computer programs and languages.[1] We have prepared this

[1] A full list of amici and their professional qualifications and contributions to the industry follows the signature page of this brief. Amici include computer scientists who developed or contributed to the development of

brief because we believe the First Circuit wisely corrected the District Court's error in overextending copyright coverage in a manner irreconcilable with the purpose of Copyright law and the nature of computer science. This brief was approved by each of the individual signatories who exercised complete control over its editorial contents. At our request, Borland helped defer the costs of printing. Both parties have consented to the filing of this brief.

Many of us filed an *amicus* brief in the First Circuit Court of Appeals, in which we discussed the copyrightability of computer languages, and urged reversal of the District Court's rulings. The District Court's opinions had generated widespread controversy and uncertainty in the computer industry because they went far beyond earlier case law, which recognized copyright coverage for the text of a computer program ("code") or for the visual, pictorial displays of the computer screen, but not for "computer processes." The District Court, however, held that functional aspects of Lotus' 1-2-3 spreadsheet program, and the program's procedures and methods of operation, were copyrightable and infringed, even though Borland copied neither the spreadsheet's programming code nor its visual displays. Specifically, the District Court held as copyrightable the program's menu commands, which are used to operate the computer, and which are used as a "macro language" for programming computer routines to execute a desired computer operation repeatedly. In protecting these computer processes, the District Court ignored the distinction between an expressive work (such as a novel or a computer program's "code"), on the one hand, and the language or medium used to express a work (such as the English language or a computer programming language), on the other hand. The First Circuit remedied that legal error by holding the menu command hierarchy to be an uncopyrightable method of operating the computer.

Not only does the First Circuit's opinion properly interpret the copyright laws, it also makes sense from a practical point of view. Fundamentally, this case directly affects the way people

(1) computer languages such as Ada, Lisp and Logo; (2) artificial intelligence; (3) robotics; (4) data encryption; and (5) other important areas of computer science.

operate machines such as computers, and the way people communicate with computers using programming languages. This *amicus* brief explains two ways in which a user or programmer would use the menu command words of Lotus 1-2-3, namely (1) as part of the human/computer interface to run the program, just as buttons or switches were historically used to run machines, and (2) as a computer language to write macro programs. Such standardized computer interfaces and computer languages are highly important. Progress in the computer industry depends upon software compatibility and the ability of computer programmers to communicate freely with each other, using common computer interfaces and computer languages. Affirmance of the First Circuit's decision will help further these important policy goals. Because the copyrightability of computer interfaces and languages is a crucial issue to the computer industry, we respectfully submit the following brief on why computer interfaces, such as the Lotus 1-2-3 menu command hierarchy and macro language, should not be copyrightable.

II. SUMMARY OF THE ARGUMENT

1. Profound historical changes have occurred over the last several decades in the way in which people communicate with machines. Buttons, switches, and electrical plugs have been replaced with computers, and a variety of human/computer interfaces. One such interface consists of the Lotus 1-2-3 menu command words and the sequencing or syntax of those words, or "menu command hierarchy." Lotus uses menu words such as "COPY" or "PRINT" in place of buttons or switches which accomplished analogous functions on earlier machines.

2. Buttons, switches, and plugs have never been considered copyrightable on their own. Replacing them with their computer equivalents should make no difference under the law. This conclusion holds true even though a "computer program" is involved. Computer interfaces or languages are not the same as the computer program itself, which is the only aspect of a computer Congress has said should be copyrightable. Instead, a computer interface is an uncopyrightable method of operating a computer, as the First Circuit correctly found here.

3. The First Circuit's opinion also recognizes the reality faced by the computer industry. Standardization of computer interfaces and languages is vitally important to this industry. Requiring each new programmer to write a new interface or language would be absurdly inefficient and would decrease, not increase, the number of creative works in the industry.

4. In this case, the Lotus 1-2-3 menu command words and the sequencing or syntax of those words also define a computer language in which users can write computer programs known as "macros." As a computer language, the Lotus menu words and order are not the same thing as a computer program. It is very important to distinguish a copyrightable computer program from the uncopyrightable language in which the program is written. Languages are the building blocks used to create works such as novels or computer programs; protecting those building blocks is quite different from protecting the expression in a work created with those building blocks. Protecting a novel written in the English language is not the same as protecting the English language itself. Before this case, no one would have seriously argued that any language could be copyrightable, let alone a computer language. There are a number of important policy reasons, discussed below, why computer languages should remain uncopyrightable.

III. ARGUMENT

A. General Discussion of Computers

The following is a basic discussion of computers helpful to understanding the issues discussed in this brief. Broadly speaking, computers have two components: *hardware* and *software*.[2] Hardware consists of the physical components used to make a computer, and includes three basic elements: the *central processing unit* ("CPU"), memory, and the input/output system. The CPU is the "brains" of the computer where the actual computing is done. On modern personal computers, the CPU is contained within a single silicon "chip" called a microprocessor. The CPU accesses various portions of the computer's memory to store and help run computer programs.

Software generally refers to computer programs, which are defined in 17 U.S.C. §101 as "a set of statements or instructions to be used directly or indirectly in a computer to bring about a certain result." Broadly speaking, there are two types of software: operating systems software and application programs. The *operating system software* is the basic set of instructions that allows the CPU to function as a computer, manages the interaction among the various hardware elements of the computer and between the hardware and the application programs, and allocates the computer's resources among other programs that might need them. The computer cannot function without operating system software. Because the operating system software must interact with other programs and with hardware components, the other programs must be compatible with the computer's operating system software, that is, they must be able to exchange information with the operating system in a precise and accurate manner. *See, e.g., Computer Associates*, 775 F.Supp. at 550. Examples of operating systems are DOS, Windows 95, and UNIX.

Once a computer has an operating system installed, it can accept additional computer programs, called *applications programs*, which perform applications such as word processing, payroll and inventory management systems, graphics, video games, or, as in this case, computer spreadsheets.

When human beings write computer programs, they generally do so in human readable code, known as *source code*. Source code is written in computer languages such as assembly

[2] Sources for this background material include: *Computer Associates Intern., Inc. v. Altai, Inc.*, 775 F.Supp. 544, 549-551 (E.D.N.Y. 1991), *aff'd.*, 982 F.2d 693 (2d Cir. 1992); F. Friedman, E. Koffman, Problem Solving, Abstraction, and Design Using C++, at 6-18, (Addison-Wesley Publishing Co. 1994), ISBN 0-201-52649-2; IBM Dictionary of Computing (McGraw-Hill 10th Ed. 1993), ISBN 0-07-031488-8(HC) (hereinafter *IBM Dictionary*); Microsoft Press Computer Dictionary, (2d Ed. 1994), ISBN 1-55615-597-2 (hereinafter *Microsoft Dictionary*).

language, BASIC, or C++. In order for the computer to understand a program, a source code program must be translated into the "1s" and "0s" which are understandable to computers. This is called *object code*, and is virtually impossible for most humans to read.

B. Human/Computer Interfaces Should Not Be Copyrightable

1. Overview of Human/Computer Interfaces

This brief now explains the two ways in which a person running the Lotus 1-2-3 spreadsheet uses the Lotus menu command words, and their order. First, the person running the Lotus program uses the menu words to operate the spreadsheet. Therefore, the words and syntax are part of what is called the *human/computer interface*, or *user interface*. Generally speaking, an *interface* is a shared boundary, or the point at which a connection is made between two different elements or functional units in a computer system so that they can work with one another. *See Microsoft Dictionary* at 218; *IBM Dictionary* at 351. The user interface consists of the graphical design, the commands, prompts, and other devices that enable a user to interact with and perform operations on a computer system or program. *See Microsoft Dictionary* at 218; *IBM Dictionary* at 724. The Lotus menu commands meet these definitions, as they are directions to the computer to "manipulate and control the program." *Lotus Development Corp.* v. *Borland International, Inc.*, 49 F.3d 807, 809 (1st Cir. 1995), Pet. App. at 4a ("*Lotus* v. *Borland*"). More specifically, the Lotus menu command words and syntax are the "specification" for the human/computer interface. This means that the human operator must use the exact words in their exact order to operate the Lotus spreadsheet. No other words or order will do.

On a steadily increasing basis, human interaction with machines is now accomplished by human/computer interfaces. The First Circuit's analogy to the video cassette recorder (VCR) is one example. *Lotus* v. *Borland*, 49 F.3d at 817, Pet.

App. at 18a. Historically, users pressed buttons to operate electronic devices such as VCRs. The first VCRs were not computer controlled; pressing the buttons operated electronic circuitry which in turn performed record, play, and other functions. Timers and clocks were largely mechanical. Modern VCRs have replaced those electrical and mechanical components with a computer. The user can still operate many of the VCR functions by pressing buttons that still say "record" or "play." Thus, the computer/human interface *looks* the same as it did 20 years ago; the only difference is that the buttons now operate a computer instead of electrical or mechanical components.

Similarly, the manual typewriter has been replaced by a computer keyboard. It still uses the same QWERTY computer/human interface, even though that interface was designed to be used with the relatively inefficient manual typewriter and is far from the optimum design for a computer keyboard. Paul A. David, "CLIO and the Economics of QWERTY," 75 *Am. Econ. Rev.* 332 (1985). Nor are contemporary computer keyboards just a collection of hardware buttons. Modern word processing programs typically have features permitting the keyboards to be redefined, in software, so that the keys can represent different letters or functions than they normally do. *See, e.g., User's Guide for WordPerfect for Windows 6.1* at 305-308 (Novell, Inc. 1994) (discussing "Keyboard Preferences" and "Keyboard Editor" features).

Computers also have progressed to the point where software "buttons" have replaced hardware buttons or switches. Lotus 1-2-3 is such an example; commands such as "COPY" or "PRINT" appear on the screen, instead of a physical button. *Lotus* v. *Borland*, 49 F.3d at 817, Pet. App. at 18a-19a. Other programs use "touch screen" menus where the user can touch a word displayed on a computer screen and operate the desired function.

Moreover, technology has progressed to the point where no manual interaction between a human being and a computer is even necessary to operate the interface. At least three companies, including Lotus' owner, IBM Corp., are selling voice dictation and voice command systems enabling the user to speak the commands to the computer instead of either typing them, using function keys, or using a mouse[3].

2. Reasons Why Copyright Protection Is Not Appropriate

Twenty years ago, a company such as Lotus could not stop a competitor from using the organization of the buttons and switches on its product merely by paying $20 and filing a two-page form with the Library of Congress. Today, it is very difficult to see why Lotus should be allowed to use the copyright laws to stop its competitors from using the same command words of the Lotus program in the same order, when those words perform precisely the same function as a set of buttons did twenty years ago. The First Circuit noted that hardware buttons would not be copyrightable simply because of the "useful article" exception in the definitions of the Copyright Act, 17 U.S.C. § 101. *Lotus* v. *Borland*, 49 F.3d at 817, Pet. App. at 19a. As scientists, it is extremely difficult to understand why replacing hardware buttons with software buttons logically should change a formerly uncopyrightable "useful article" into something that is copyrightable, just because a "computer program" is involved. A human/computer interface, or a computer language, cannot be viewed in the same manner as any ordinary "literary work" exempt from the useful article exception of § 101. Someone can read an ordinary literary work, such as a novel or a play, and enjoy it for what it is worth. Someone can also read the text of a computer program written in source code as a work of authorship and gain some technical knowledge in the same way as one might gain such knowledge by reading a book about computers.

[3] In addition to IBM, the companies include at least Dragon Systems, Inc. of Newton, Massachusetts, and Kurzweil Applied Intelligence, Inc. of Waltham, Massachusetts. *See* Wayne Rash, Jr., *Talk Show: Voice-recognition Technology for the PC Lets Your Voice Run the Show*, PC Magazine, December 20, 1994, at 203, 205; Richard A. Shaffer, *Computers With Ears*, Forbes, September 12, 1994, at 238.

Computer interfaces are fundamentally different from any of these. They are singularly functional and utilitarian in nature. One does not normally "read" a set of commands such as "COPY" or "PRINT," or the syntax or structure of those commands, as one might read a book about computers. Rather, the purpose of human/computer interfaces is to operate functions of the computer machine itself, and to use the computer.

Because of the nature of human/computer interfaces, there are a number of policy reasons why they should not be the subject of copyright protection. First, factors such as software compatibility, interoperability, and hardware portability are extremely important in the computer industry. Computers essentially have replaced the methods we used to operate machines in the past (hardware buttons or switches) with identically functioning buttons that instead operate software. In some instances, the physical switches appear the same to the user; in other instances, they are words on a computer screen.

This trend will only accelerate in the future. With computing hardware becoming increasingly inexpensive, software "virtual machines" have replaced electronic hardware in entire sectors of the economy. (A software "virtual" machine is one where the functions of electronic or mechanical components are primarily performed by software.) The ramifications of this computer revolution are enormous. The number of different kinds of computer/human interfaces have dramatically increased, both in sheer quantity as well as in the many different ways in which humans now interact with computers on a daily basis. Voice commands, and doubtless other non-physical methods of communication not presently imaginable, will increasingly replace hardware and even software buttons and switches.

The ability of programmers to write software programs that can be understood by other computers worldwide, on many types of computer platforms, is crucially important. The key point is that once an interface is created, it can become an industry standard, and the interface becomes functionally embedded not just in the original work, but in programs, and in the minds of programmers. The QWERTY typewriter keyboard and the positioning of automobile controls such as the gas and brake pedals and gear shift patterns are hardware examples of such standards; common computer languages such as BASIC and C++ are examples in the software industry.

Once a standard interface has been created, it would be absurdly inefficient to require all subsequent authors of computer software to create their own interface with each new program. Imagine if each author of a novel were required to invent a new language because a prior author had a copyright in the English language. This would lead to *less* works of authorship being created, rather than encouraging the proliferation of works as the copyright laws were intended to do. *See Fogerty* v. *Fantasy, Inc.*, 510 U.S. ___, 127 L.Ed.2d 455, 465, 114 S.Ct. 1023, 1029 (1994) (noting the ultimate aim of the Copyright Act is "to stimulate artistic creativity for the general public good").

Thus, granting copyright protection in computer interfaces would allow the creator or "first comer" to create a computer interface and use it to lock up the technology and marketplace for 75 years, even if the interface and technology was unpatentable. This would create enormous barriers to entry in a market where the free exchange of ideas is essential. *See, e.g. Computer Associates International, Inc.* v. *Altai, Inc.*, 982 F.2d 693, 712 (2d Cir. 1992), citing P. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs,* 41 Stanford L. Rev. 1045, 1082 (1989). This effective monopoly on a computer interface would inhibit third-party innovations, particularly given the long term of copyright. *Cf. Bonito Boats, Inc.* v. *Thunder Craft Boats, Inc.*, 489 U.S. 141, 146, 103 L. Ed. 2d 118, 131, 109 S.Ct. 971, 975 (1989) ("imitation and refinement through imitation are both necessary to invention itself and the very lifeblood of a competitive economy."). If the interface was copyrightable, the interface owner also would "lock up" an installed base of users accustomed to that interface.

C. Computer Languages Should Not Be Copyrightable

1. Overview of Computer Languages

This portion of the brief discusses the second way a user uses the Lotus menu words, as a computer language to write macro programs.[4] The Copyright Office has defined a computer language as "a programming language used by a programmer for writing a computer program."[5] A commentator similarly has described a computer language as "a code that both the computer operator and the computer understand. The operator uses the language to express what is to be done, and the computer understands the language and performs the desired actions."[6] A computer language thus is similar in many respects to an ordinary language such as English. Like an ordinary language, it consists of basic building blocks of communication. It provides rules (grammar and syntax) for the combination of words or terms to form expressive works that can be understood by their authors and other programmers.

[4] Lotus has denominated the Lotus macro language as the method by which the macro programs can interact with the Lotus 1-2-3 program. *See* Declaration of Vern L. Raburn, ¶ 18 and Exhibit B, J.A. 530, 535. In this respect the Lotus macro language acts as a program-to-program interface specification. We understand that other *amici* are discussing the uncopyrightability of program-to-program interfaces, and therefore will not discuss it in this brief.

[5] Compendium II of Copyright Office Practices, the Library of Congress, §326 at 300-32 (1984).

[6] E. Lowry, *Copyright Protection for Computer Languages: Creative Incentive or Technological Threat?*, 39 Emory L.J. 1293, 1298 (1990) (citation omitted) (hereinafter "*Lowry*"). Another commentator summarized a number of definitions of computer language as follows:

> A computer programming language is a formal system of expression including:
> (1) a set of vocabulary elements;
> (2) a set of syntax rules for combining vocabulary elements into statements; and
> (3) the assignment of meaning to statements that properly combine vocabulary elements in accordance with the syntax rules.

R. Stern, *Copyright in Computer Programming Languages*, 17 Rutgers Computer & Tech. L.J. 321, 327 (1991) (hereinafter "*Stern*").

Unlike ordinary languages, computer languages serve an important additional purpose. They not only permit communication between different computer programmers, but also allow programmers to communicate with a computer and tell it what to do. They are highly utilitarian languages that tell the computer how to implement the desired functions, operations, and results desired by the program. In short, they permit computers to "talk" to each other and to exchange that which will be understood by the operators of the computer.

A computer language is fundamentally different than a program written in that language. The rules for the language consist of definitions and syntax, usually written in alphabetical or some other logical order. However, a program uses selected words of the language in the order appropriate for the function being performed. This concept is illustrated by the following example using the BASIC programming language. (The commands are listed in the alphabetical order used in the BASIC manual, and are only the small group of the many available BASIC commands which are actually used in the sample program, plus a few others.)

COMPARISON OF BASIC COMMANDS
WITH A PROGRAM WRITTEN IN BASIC

```
BASIC
Commands     Sample Computer Program
DATA         10 PRINT ''This program adds.''
END          20 PRINT ''Type a number.''
GOTO         30 INPUT A
IF/THEN      40 PRINT ''Type another number.''
INPUT        50 INPUT B
LET          60 LET C = A + B
PRINT        70 PRINT ''The sum is'', C
READ         80 END
```

The above program asks the user to type in two numbers; it then computes the sum of the two numbers and prints the sum. In the terminology of 17 U.S.C. § 101, the eight lines numbered

10 through 80 are the "set of statements or instructions" of the program; the printed sum is the "certain result" of the program. Thus, the program on the right is different from the list of commands on the left. (While the "certain result" in this example is a number, in other programs the certain result can be other matters such as a still or video image, audio recordings, or, as in this case, command words such as "COPY" or "PRINT").

2. The Lotus Macro Language

In this case, the Lotus 1-2-3 menu command words and the sequencing or syntax of those words define a computer language. Lotus 1-2-3 allows users to write their own computer programs, called "macros," using this programming language. *See Lotus* v. *Borland*, 49 F.3d at 809, Pet. App. at 4a. The District Court defined a "macro language" as "a feature by which a user may define a very short keystroke sequence as equivalent to a longer keystroke sequence."[7] As explained by the District Court, the Lotus 1-2-3 macro language permits a user to write a series of executable computer operations (the "macros") by using a language of special abbreviated programming commands (such as "R", "F", or "C"), and special symbols (such as "/" or "{?}"). *Borland IV*, 831 F.Supp. at 226-27, Pet. App. at 31a-32a. Thus, the 1-2-3 macro language is a set of rules for writing a computer program (the macro), and meets the definition of a computer language discussed above. Once users become reasonably adept with the operation of the spreadsheet program, they will often write macros to save time and effort.

[7] *See Lotus Development Corp.* v. *Borland International, Inc.*, 799 F.Supp. 203, 206 (D.Mass. 1992), Pet. App. at 110a. (hereinafter "*Borland II*"). This brief also refers to four other relevant opinions by the District Court: *Lotus Development Corp.* v. *Paperback Software International*, 740 F.Supp. 37 (D.Mass. 1990) (hereinafter "*Paperback*"); *Lotus Development Corp.* v. *Borland International, Inc.*, 778 F.Supp. 78 (D.Mass. 1992) (hereinafter "*Borland I*"); *Lotus Development Corp.* v. *Borland International, Inc.*, 831 F.Supp. 202 (D.Mass. 1993) (hereinafter "*Borland III*"); *Lotus Development Corp.* v. *Borland International, Inc.*, 831 F.Supp. 223 (D.Mass. 1993) (hereinafter "*Borland IV*").

Lotus has argued that the menu command hierarchy is a "computer program," and hence protectible under the copyright laws. For the reasons discussed above, this is not so. The words and syntax of the hierarchy can be used to write copyrightable computer programs, *i.e.* the macros. However, when the Lotus menu words and syntax are used in this manner, they are a language, not a program. They are the words from which the macro programmer creates the macros, which are the "set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result" under 17 U.S.C. §101. In this respect, the Lotus menu words and syntax are similar to the words and syntax of a "national language," like English or French, which can be used to create copyrightable works of literature, and are similar to the BASIC language example noted above.

Again, this can be illustrated by an example comparing some of the Lotus command words with a macro written using those words. In the following example, the left column lists the "File" and "Data" commands used in the first two lines of the sample macro, in the order in which they appear in Lotus 1-2-3, as shown at J.A. 900-906. The macro example on the right is taken from the Declaration of Judith S. Olson, J.A. 508-09.

COMPARISON OF LOTUS 1-2-3 COMMANDS WITH A LOTUS MACRO PROGRAM

```
Lotus
Commands                      Sample Macro

''/F''ile Commands            /fit{?}~
.  .  .                       /dprfcfe{?}~
''I''mport                    i.{end}{down}~
''T''ext                      o{right}~g
.  .  .
''/D''ata Commands
.  .  .
''P''arse
''F''ormat Line
 ''C''reate
 ''E''dit
.  .  .
''R''eset
```

As Dr. Olson's declaration explains (J.A. 508-09), the macro program is used to extract data from another file, such as a database, and re-format it for use in a spreadsheet. The first line of the macro, for instance, opens a file ("f"), and imports it ("i") from the database as text ("t"). (Unlike BASIC, which as shown in the previous example uses entire English-looking words such as "PRINT," the Lotus macro language often uses just the first letter of the relevant command. *See Borland IV*, 831 F.Supp. at 226-27, Pet. App. at 31a.) The four lines of the macro are the "set of statements or instructions" of the program; the re-formatted data file is the "certain result" of the program. As was the case with the previous example, the program on the right is different from the list of commands on the left.

3. Reasons Why Copyright Protection Is Not Appropriate

It is very important to distinguish a copyrightable computer program from the uncopyrightable language in which the program is written. Copyright law was intended to cover the expressive nature of an author's works, and to encourage their creation. However, it was never meant to cover the basic building blocks, or language, used to create the works.[8] Protecting the expression in a work is quite different from protecting the building blocks used to create the work. Protecting a novel written in the English language is not the same as protecting the English language itself. Nor is protecting a work of art such as a painting the same as protecting the various colors of paint used to create the painting. As the above examples make clear, a computer language is not the same as the "computer program" defined in 17 U.S.C. § 101. Accordingly, before this case, no one would have seriously argued that any language could be copyrightable, let alone a computer language.[9]

There also are a number of good policy reasons why computer languages should not be copyrightable. The reasons discussed above why copyright protection is not appropriate for human/computer interfaces generally also apply to computer languages. Standard languages are very important for a wide variety of reasons: promoting ease of use by programmers, avoiding lock-ins of the user base and the technology by the

[8] *Lowry* at 1315; I. Paul Goldstein, *Copyright Principles, Law and Practice*, §1.2.2.4 at 16 (1989). See also *Stern* at 349, 364.

[9] The District Court implicitly granted copyright protection to the Lotus menu command hierarchy as a computer language. *Borland IV*, 831 F.Supp. at 229-230, 234, Pet. App. at 37a-39a, 47a; see also *Borland II*, 799 F.Supp. at 213-14, Pet. App. at 122a-125a. The District Court seemed to think that languages generally were copyrightable. *Paperback*, 740 F.Supp. at 72, Pet. App. at 243a-244a. See *Stern* at 323-24, 330, noting that *Paperback's* statement to this effect was unprecedented. This was one of the reasons why the District Court's opinions generated such controversy in the computer industry and why a number of us filed an *amicus* brief in the First Circuit criticizing the District Court.

first company on the market to write a language, providing incentives to create new works using a standard language, and preventing the disincentives that arise from requiring each competitor to create its own language.

The First Circuit acknowledged the realities faced by the computer industry. The majority opinion recognized the importance of program compatibility and the importance of allowing users to run programs such as macros on different platforms. The majority opinion concluded that "forcing the user to cause the computer to perform the same operation in a different way ignores Congress's direction in §102(b) that 'methods of operation' are not copyrightable." *Lotus* v. *Borland*, 49 F.3d at 817-18, Pet. App. at 20a. Judge Boudin's concurring opinion also correctly observed how providing a monopoly of a computer command structure would make the users "captives" of the first provider of the program, not because of any creative effort or investment by that programmer, but because of the "investment in learning made by the users." *Id.* at 821, Pet. App. at 26a-27a.

Providing copyright protection for a computer language, such as the Lotus macro language, would have an additional undesirable effect. Such protection would extend far past the computer language itself. For example, granting copyright coverage for computer languages would allow the language owners to claim ownership of all programs written in the languages as derivative works. If languages were copyrightable, works created using the language would be derivative works of the language.[10] In that event, the computer language owner could

[10] *See* the definition of derivative work in 17 U.S.C. §101:
A 'derivative work' is a work based upon one or more preexisting works, such as a translation, . . . abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations or other modifications . . . is a 'derivative work'.
Under this statute, a program is a work "based upon" the preexisting language; the language is "recast, transformed or adapted" into the program, which in any event is an "elaboration or other modification" of the language.

argue that he owned all programs based on his language.[11] This objectionable result is a natural consequence of confusing the expressive work itself with the language used in creating the expressive work, and permitting copyright coverage for both.

D. The Menu Command Words Need Not Be Spelled Out in the Code

Finally, as computer scientists we wish to respond to one point in the Lotus brief which we believe to be incorrect or potentially misleading. Lotus states that the actual menu command words are "spelled out, in text, in the program code." Lotus Br. at 7. While the record cites at this point are to a *Borland* data file which needed to recognize the 1-2-3 command words and order to operate, Lotus' statement incorrectly implies that the words did in fact appear in the *Lotus* code. Since the Lotus code was never introduced into evidence, J.A. 300, one cannot conclude that the menu words actually appeared in the Lotus code. From a programming standpoint, there are a number of ways the Lotus code could have implemented the menu functions without spelling out the words in the code.

For example, an extremely common programming technique used for programs with international sales (such as word processors or database programs) does not spell out the menu command words in the program code. The menu words are instead contained in a data file which is separate from the code used to run the program. This makes the menu words easy to change when the program is sold to non-English speaking users, since the words are isolated in one location instead of being scattered throughout thousands or millions of lines of code. It also has the advantage of permitting users to customize the menu command words if they want to. While we do not know if this or a similar technique was actually used in Lotus 1-2-3, since the Lotus code was never placed into evidence it cannot

[11] *See* 17 U.S.C. §106(2).

be assumed that such a technique was *not* used. Thus, one cannot state, as does Lotus, that its menu words were "spelled out" in its code.

IV. CONCLUSION

As computer scientists, we believe that the majority and concurring opinions of the First Circuit correctly recognized the danger in extending copyright protection to the specific type of human/computer interfaces and computer languages at issue in this case. For people in our computerized society, it is vitally important to be able to communicate freely with computers, without threat of litigation. Computer interfaces and languages are meant for everyone to use. They were never meant to enable their creator to monopolize methods of operating computer-based machines. The District Court's decisions lost sight of the importance of being able to use common, standard computer interfaces. The First Circuit correctly understood the issues involved here, and remedied the District Court's error.

For the foregoing reasons, the judgment of the First Circuit should be affirmed.

Respectfully submitted,

RON KILGARD
(Counsel of Record)
KARL M. TILLEMAN

Dalton, Gotto, Samson & Kilgard, P.L.C.
Suite 900
3101 North Central
Phoenix, AZ 85012
Phone: (602) 248-0088

Attorneys for Amici
Listed in Appendix A

December 1995

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A
LIST OF SIGNATORIES TO THE BRIEF

[In Alphabetical Order]

1. Harold Abelson

Harold Abelson is Class of 1922 Professor of Computer Science and Engineering in the Department of Electrical Engineering and Computer Science at MIT. He was recipient in 1992 of the MIT Bose Award (School of Engineering teaching award) and is a Fellow of the IEEE. In 1992, Abelson was designated as one of MIT's six inaugural MacVicar Faculty Fellows, in recognition of his significant and sustained contributions to teaching and undergraduate education. He has made many contributions to the development of the Logo educational computing language. He directed the first implementation of Logo for the Apple II, which made the language widely available on personal computers beginning in 1981. Together with Prof. Gerald Sussman, Abelson developed MIT's introductory computer science subject, "Structure and Interpretation of Computer Programs," a subject organized around the notion that a computer language is primarily a formal medium for expressing ideas about methodology, rather than just a way to get a computer to perform operations. This work has had a world-wide impact on university computer-science education.

2. William B. Ackerman

Designer/implementor of one of the first capability-based multiprocessing operating systems, 1968; Author of the extremely popular ISPELL spelling checker, 1978; Co-author of the VAL dataflow programming language, 1982; Co-discoverer of the "Brock-Ackerman Anomaly" in denotational semantics of nondeterministic message-passing systems, 1984.

Currently at Hewlett-Packard Laboratories, Cambridge, MA.

3. Dean Anderson

I am President of the League for Programming Freedom, an organization founded to fight user interface copyrights and software patents. My software background includes 9 years of programming and system administration experience working on a wide variety of computer systems, languages, and operating systems for companies such as Charles Stark Draper Laboratory, Open Software Foundation, Kendall Square Research, and Hitachi Computer Products. I have contributed to the development of OSF/Motif, OSF/1 and other projects on systems ranging from mainframes to PC's. I have worked with a variety of operating systems from many flavors of unix to IBM MVS and VM to Microsoft DOS, Windows, and NT. I have programmed in a variety of languages from scheme, C, PLI, to TCL and Perl. In the last 4 years, I have worked on implementing a wide variety of networks including FDDI, CDDI, 100BaseT, Token Ring, Ethernet, Frame Relay, Dialup, supporting several protocols such as IP, Novell, and Appletalk. Currently, I am also President of Plain Aviation, Inc., a company started in 1992 which offers aviation and engineering services filling in corporate infrastructure gaps and recently offering Systems and Network Consulting, Custom solutions, and Co-sourcing services.

4. Judy Anderson
Harlequin Inc.
Cambridge MA 02142

Judy Anderson has been working as a programmer in the computer industry for over a decade. Her work has primarily been programming in Lisp and C, particularly doing low level implementation of Lisp programming environments. Of late, she has been interfacing between the Lisp system sold by Harlequin and various database products sold by other vendors.

5. Phil Bagley

Graduate of MIT in Electrical Engineering, Master of Science 1951. Thesis: High-speed digital machines for information searching. MIT Digital Computer Lab and MIT Lincoln Lab, 1951-1959, research associate. MITRE Corporation, 1959-1964, technical staff member.

Assistant Professor of Computers and Information Science, Temple University, 1978-1984.

6. Brian Bartholomew

I am a professional systems administrator for Unix, Windows, and Macintosh computers. I make my living extending the abilities of users by connecting and integrating disparate applications and operating systems in ways their manufacturers did not plan for.

7. Karl Berry

I am a member of the technical staff at Interleaf Inc., an Adjunct Research Associate at the University of Massachusetts at Boston, co-author or co-editor of several technical books published by Addison-Wesley and Kluwer, and maintain various freely available software packages.

8. Ethan Bradford

Education: BS, Mathematical Sciences, Stanford U., June 1981 (with Honors and Distinction); MS, Computer Science, Stanford U., June 1982; MS, Physics, U. of Washington, June 1991; PhD, Physics, U. of Washington, expected June 1992

Full time employment: Full Staff Member (doing programming and systems analysis), MIT Lincoln Laboratory, June 1982-August 1989

9. Seth Breidbart
Ph.D., Computer Science, Yale 1978
Co-author of the fundamental papers on Quantum Cryptography

10. Robert E. Bruccoleri, Ph.D.
Principal Scientist

Department of Macromolecular Modeling
Bristol-Myers Squibb Pharmaceutical Research Institute

Primary author of the program CONGEN, a mulitfunctional molecular modeling package, which is used to predict the structure of proteins by homology. He obtained his B.S. in Computer Science from the State University of New York at Stony Brook, his M.S. in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology, and his Ph.D. in Biochemistry and Molecular Biology from Harvard University.

11. Matthias Bruestle
Student of Chemistry at the University of Erlangen
PGP — Administrator of the KommunikationsNet Franken e.V.

12. Steve Byrne

Mr. Byrne is a graduate of Carnegie-Mellon University, and has been in the computer industry for 15 years. He has worked with everything from operating systems, to expert systems, to graphical user interface based end-user applications. He has developed software on everything from 8 bit microprocessors to mainframes, and has over 20 years of experience working with computers, and is regarded both as a senior design and architectural contributor, as well as an experienced user interface designer.

Mr. Byrne is the creator of the GNU version of the Smalltalk programming language. He has also created implementations of several other programming languages, as well as several utility programs and packages. His current interests are object oriented systems, distributed programming, programming languages, and graphical user interface design and implementation.

13. James Carlson

I am currently a Principal Software Engineer for Xylogics, Incorporated, and I develop communications software

for the Annex line of terminal and communications servers, which are used by many Internet dial-up sites and universities as the interface between the network and the telephone system.

14. Mike Coleman

Mike Coleman holds a Master's degree in Computer Science from UCLA and has a number of years of industrial software development experience, including programming language and user-interface design. He is presently at the Center for Telecomputing Research at the University of Missouri-Kansas City.

15. Richard H. Crawford

Currently a Researcher in Computer Security at the University of California, Davis. His three years of work experience with Hewlett-Packard as a Software Development Engineer involved designing user interfaces to database manipulation programs.

Crawford received a B.S. in Mechanical Engineering from UC Berkeley in 1979, followed by a M.S. in Computer Science from UC Davis in 1991. After completing his M.S. Thesis, "Topics in Behavioral Modeling and Event-Based Debugging", Crawford continued to pursue several strands of this research.

16. Pavel Curtis

Dr. Pavel Curtis has been a member of the research community at the Xerox Palo Alto Research Center since 1983, during which time he has worked on aspects of the Smalltalk-80, Interlisp-D/Xerox Lisp, and Cedar programming environments and on other projects mostly related to the design and implementation of programming languages, including leadership of the SchemeXerox project exploring large-scale software development in the Scheme programming language. He is the founder and chief administrator of LambdaMOO, one of the most popular recreational social virtual realities on the

Internet. His current work centers on the Network Places project, investigating the implementation, applications, and implications of systems that allow multiple simultaneous users to communicate and interact in pseudo-physical surroundings.

17. Josef Dalcolmo

I am a Senior Development Engineer (holding a Ph.D. in Physics) at the University of California at Berkeley. I have been working for the Austrian Academy of Sciences, the European Space Agency and for the last six years for the University of California at Berkeley, Space Sciences Laboratory.

18. Hugh Daschbach

Hugh Daschbach has been a professional programmer since 1973. He has worked on numerous projects for both computer users and vendors. Now employed as an independent software consultant, he has worked on projects for Digital Research, IBM, Xerox, FileNet Corporation, and the Aluminum Company of America.

19. L. Peter Deutsch

Dr. L. Peter Deutsch received the Ph.D. in Computer Science from U.C. Berkeley in 1973. Prior to the Ph.D. he was one of the key designers and implementors of the SDS 940 time-sharing system, the first commercial, general-purpose time-sharing system using paging hardware. In his subsequent 13 years at Xerox PARC, where he attained the position of Research Fellow, he was one of the key designers and implementors of the research prototype of the Interlisp-D system; a significant contributor to the design of the Cedar Mesa language and the Smalltalk-80 programming environment; and the principal creator of PS, the first high-performance implementation of the Smalltalk language and programming environment on microprocessor-based hardware. From 1986 to 1991, Dr. Deutsch was Chief Scientist at ParcPlace Systems, where he was a principal designer of a high-performance

Smalltalk implementation that is also highly portable across processors, operating systems, and window systems. From 1991 to early 1993, Dr. Deutsch held the position of Sun Fellow at Sun Microsystems, where he helped define future corporate strategy and technology in a variety of areas. In 1993, Dr. Deutsch founded, and from 1993 to 1995 served as President of, Artifex Software Inc., which commercially licenses a highly portable high-performance implementation of the PostScript language developed by Dr. Deutsch that is also available for non-commercial use from the Internet under the name Ghostscript. Dr. Deutsch was a co-recipient of the ACM Software System Award in 1993, and in 1993 was also named a Distinguished Alumnus of the U.C. Berkeley Computer Science program. Dr. Deutsch is a member of ACM, IEEE, CPSR, and the League for Programming Freedom.

20. Antonio J. De Vido

Systems Administrator, Sanwa Financial Products, L.P. B.S. Computer Science, Yale University 1993.

21. Robert B. K. Dewar

Professor of Computer Science and Associate Director of the Courant Institute of Mathematical Sciences at New York University. Active in operating systems and programming languages design for thirty years. Principle designer and author of the SPITBOL system. Chairman of IFIP WG2.1. Member of ISO standardization committees. Author of "Microprocessors: A Programmer's View", McGraw Hill 1990. Active in the design and development of the Ada language. One of the principle authors and designers of the GNAT system, a completely free software implementation of Ada 95 in current use at thousands of sites world wide. President of Ada Core Technologies, a company dedicated to Free Software and the encouragement of the use of Ada 95.

22. Mark-Jason Dominus

I was a systems programmer for the University of Pennsylvania Department of Computer and Information Sciences from 1990-1994, when I left to help start 'Pathfinder', Time-Warner's Internet information service. I now run Plover Systems, which provides consulting services to people who want to set up Internet-based information services.

23. Lester Earnest

Education:
B.S. Electrical Engineering, Caltech, 1953
M.S. EE/Computer Science, MIT, 1960
Professional experience:
1988-present: Consultant, Los Altos Hills, CA
1985-1988: Stanford University, Stanford, CA; Associate Chairman of Computer Science Department
1980-85: Founding President of Imagen Corporation, Santa Clara, CA, which introduced the first desktop publishing systems using laser printers.
1965-80: Executive Officer, Artificial Intelligence Laboratory, Stanford University, Stanford, CA.
1963-65: Department Head, Information Systems Dept., Mitre Corp., Washington, DC
1958-63: Subdepartment Head, Intelligence Systems Dept., Mitre Corp., Bedford, MA.
1956-57: Member, Technical Staff, MIT Lincoln Laboratory, Lexington, MA.

Invented the first computer spelling checker (1967). Created the first pen-based computer system that reliably recognized cursive handwriting (1961). Created the first FINGER program (1973), which has since become a standard utility on most computer systems on the Internet. Was a member of the technical committee that evaluated initial bidders on the creation of ARPAnet, which became the Internet. Organized and led the first successful robotics project at Stanford, which employed a TV camera and

mechanical arm (1966-68). Later initiated a visually-guided vehicle project (Stanford Cart, 1969), which led to a number of Ph.D. dissertations.

24. Paul Eggert
Director, R&D
Twin Sun Inc.
El Segundo, CA

Paul Eggert has directed software research and development projects at Twin Sun since 1989 in the areas of networked multimedia, computer supported cooperative work, and software reliability. From 1986 to 1989, Eggert was Senior Advisor to Management at Unisys Corporation, mostly working in the area of software verification. From 1983 to 1986, he was technical director and then Chief Scientist at Silogic Inc, directing development of logic programming software tools. From 1980 to 1983, he was Assistant Professor of Computer Science at the University of California, Santa Barbara.

25. Thomas G. W. Epperly

Computational Chemical Engineering. Education: BS Chemical Engineering, Carnegie-Mellon University, 1988; PhD Chemical Engineering, University of Wisconsin-Madison, 1995. Honors & Awards: Computational Science Graduate Fellowship (U.S. Dept. of Energy) 1991-95. Experience: Research Associate, Imperial College, England.

26. Michael Ernst

Lecturer at Rice University. He has performed research in the areas of artificial intelligence, parallel algorithms, compilers, cryptography, intellectual property, compilers, and static analysis. He has a Bachelor's and Master's degree from MIT.

27. Craig A. Finseth

Author of the main book on how to write text editors: "The Craft of Text Editing, or, Emacs for Modern Times."

Also, author of several programs distributed under GNU CopyLeft including Freyja (an Emacs for very small machines) and Loki (a calculator for the same). I am the manager of the data network for the State of Minnesota (~53,000 computers), an area in which it is crucial that users have access to multiple vendors that implement the same interfaces.

28. Simson L. Garfinkel
Author, Practical UNIX Security (O'Reilly & Associates, 1991)
PGP: Pretty Good Privacy (O'Reilly & Associates, 1994)
NeXTSTEP Application Programming (1993)

29. Daniel E. Geer, Jr., Sc.D.

Head of engineering for special projects at Open Market, Inc., a market leader in electronic commerce technology with corporate headquarters in Cambridge, Mass. Dr. Geer has been a successful entrepreneur in network security and distributed systems management, culminating in a successful sale of his own company to OpenVision Technologies, where he subsequently served as Chief Scientist, V.P. of Technology, and Managing Director of their security consulting services. Previously, he served as Technical Director, Innovation Technology Resource Center, Digital Equipment Corporation, and was Manager of Systems Development for MIT's Project Athena where he was the responsible manager for all technical development, including X, Kerberos, and all other aspects of the Project Athena Network Services System. He holds a Bachelor of Science in Electrical Engineering and Computer Science from MIT, and a Doctor of Science in Biostatistics from Harvard University, and was deeply involved in medical computing for fifteen years.

30. Howard Goldstein

Mr. Goldstein is President and CEO of InfoMotion Inc., a Florida software engineering firm that develops software

for the wireless data-communications industry. He graduated *cum laude* from Stetson U. College of Law, and is a member of The Florida Bar.

31. Allan Gottlieb
Professor of Computer Science, New York University
Director, NYU Ultracomputer Research Laboratory
Co-editor, Journal of Parallel and Distributed Computing

32. Paul Haahr
Employment: Harlequin Inc, senior software engineer, 1994-present.
Kaleida Labs, Inc., senior software engineer, 1993-94.
Adobe Systems Inc., computer scientist, 1990-93.
Polygen, Inc., 1987, systems administrator.
Oracle Systems, Inc., 1986-87, member of technical staff.
Education: Princeton University, AB in computer science, summa cum laude, 1990.
Published work: Contributor to Dylan Language and Reference Manual, Apple Computer, 1995. "Es: A shell with higher-order functions," co-authored with Byron Rakitzis, Winter 1993 Usenix conference proceedings. "Montage: Breaking windows into small pieces," Summer 1990 Usenix conference proceedings.

33. Jeffrey R. Hagen

Currently Senior Scientific Programmer Analyst with the National Radio Astronomy Observatory in Tucson Az.

Past five years responsible for the real-time control system at the Twelve Meter Radio Telescope on Kitt Peak. Wrote the 'RAMBO' interpretive C language which we use to define observing modes at the twelve meter. Wrote the telescope operators interface and well as others.

34. Darrel Hankerson
Associate Professor Department of Discrete and Statistical Sciences

120 Mathematics Annex
Auburn University

35. Diane Harrison
Business programmer at Triad since 1979

36. Brian Harvey
Lecturer, Computer Science Division, University of California at Berkeley

I am the author of several books about computer science, and the chief implementor of Berkeley Logo, a freeware version of the Logo programming language.

37. Jeff Hay

Computer Science undergrad at New Mexico Tech. Has several small programs in the public domain/shareware markets.

38. Christopher T. Haynes

Christopher T. Haynes is Associate Chair for Education and Associate Professor in the Computer Science Department of Indiana University at Bloomington, where he directs the Scheme Educational Infrastructure project funded by the National Science Foundation. He chaired the IEEE Scheme Working Group that developed the IEEE and ASCII Scheme standard and has been from its inception a member of the Scheme report author's group. Professor Haynes is co-author of the MIT Press/McGraw Hill text "Essentials of Programming Languages", which uses Scheme throughout. Professor Haynes has authored over a dozen programming language research papers. As well as teaching at Indiana University since 1982, he has taught graduate courses at MIU in Fairfield, Iowa, IISc in Bangalore, India, and ITESM in Monterrey, Mexico, and has consulted with industry. He received a PhD degree in computer science from the University of Iowa in 1982 and a BA degree in mathematics from Lehigh University in 1973.

39. Richard F. Hilliard, II

Lead systems architect, at the MITRE Corporation. Active in ISO, ANSI and U.S. Department of Defense Standardization activities. Member of the Ada 9X Mapping/Revision Team.

40. Michael D. Hirsch
Assistant Professor of Mathematics and Computer Science
Emory University
AB: Princeton 1984; PhD: Berkeley 1990
NSF Postdoc: 1990-1993

41. Christian D. Hofstader

Chris Hofstader is a co-founder, director and the primary organizer of the League for Programming Freedom. He is also a programmer and has contributed to many popular PC products including: The Bank Street Writer, Portfolio's Dynodex and WordPerfect Works for Windows. His recent projects include the design of user interfaces for blind and low vision users of personal computers.

42. Max Hyre
Staff Engineer
General Signal Networks
Shelton, Connecticut 06484

I have been a programmer of embedded communications systems for 17 years, and as such my work has relied critically on the availability of multiple programs presenting the same user interface (such as text editors & word processors, compilers, and debugging programs).

43. Philip Jensen

I have been working in computer software since high school days. I hold the degree of Bachelor of Arts in Applied Mathematics, granted by Harvard College. I am currently employed by CompuServe, Inc.

44. Jim Johnson

I have been a published computer industry columnist since June of 1990, covering career issues and industry trends. I'm also a Certified Personnel Consultant (CPC), and have served as a personnel consultant to the computer industry since 1987.

45. Phil Karn

I am an engineer in the communications industry with 16 years of professional experience. I have written a computer networking software package that is very widely used to provide low-cost access to the Internet over ordinary telephone lines. I am currently employed in the digital wireless communications industry and reside in southern California.

46. John Kohl
Systems Software Programmer, 10 years experience.

Mr. Kohl has authored technical papers on computer security and file system storage, and many freely-available software packages including hardware device driver modules, disk filing system modules, and network authentication programs. He holds a MS in computer science from the University of California, Berkeley, and a BS in computer science from the Massachusetts Institute of Technology.

47. Markus Krummenacker
Software Consultant

1994-1995: Director of Research, Nanothinc, San Francisco, California. Key work in overall layout of WWW server and company strategy. 1992-1993: Research Associate, Institute for Molecular Manufacturing, Palo Alto, California. Analysis of Molecular Building Blocks, and writing the CavityStuffer Program (in CommonLISP). 1990-1992: Visiting Researcher, Dept. of Molecular and Cell Biology, University of California at Berkeley. Isolation via PCR, and sequence determination of homologous genes for the Ferric Uptake Repressor in bacteria.

48. Andrew Kuchling

Andrew Kuchling is a native of Canada, currently working as a programmer for a Montreal software firm that is a leader in the field of corporate calendaring software. His interests include cryptography, novel programming languages, and general network applications, such as the World-Wide Web. Among other things, he maintains the major toolkit of cryptography extensions for the Python programming language.

49. Phil Laird

Dr. Philip Laird is a Computer Scientist with the Computational Sciences Division at NASA's Ames Research Center in Mountain View, California.

50. Charles Landau
Tandem Computers
Cupertino, CA 95014
BSEE, MIT, '71; MSEE, MIT, '72
Founder, Key Logic Inc. (a software company)
Founder, MACS Lab, Inc.
Architect of the KeyKOS operating system.
Author of several published papers on KeyKOS and other subjects.
Computer programmer from 1968 to the present.

51. Emery W. Lapinski
I hold a BA in CS from NYU and have been professionally designing and implementing GUIs for applications used in the Banking and Telecommunications industries for the past few years.

52. James E. Leinweber

I have used computers for over 23 years, and have been a full-time professional computer programmer, analyst and administrator since 1984. I currently work for the Wisconsin State Laboratory of Hygiene, a public health

Lab at the University of Wisconsin in Madison. I hold a Master's degree in Mathematics from Michigan State University, and am a member of Phi Beta Kappa, Computer Professionals for Social Responsibility, and the League for Programming Freedom.

53. John McCarthy

John McCarthy is Professor of Computer Science at Stanford University. He has been interested in artificial intelligence since 1948 and coined the term in 1955. His main artificial intelligence research area has been the formalization of common sense knowledge. He inv nted the LISP programming language in 1958, developed the concept of time-sharing in the late fifties and early sixties, and has worked on proving that computer programs meet their specifications since the early sixties. He invented the circumscription method of non-monotonic reasoning in 1978.

His main research (1995) is formalizing common sense knowledge and reasoning. His most recent paper is on formalization of context.

McCarthy received the A. M. Turing award of the Association for Computing Machinery in 1971 and was elected President of the American Association for Artificial Intelligence for 1983-84 and is a Fellow of that organization. He received the first Research Excellence Award of the International Joint Conference on Artificial Intelligence in 1985, the Kyoto Prize of the Inamori Foundation in November 1988, and the National Medal of Science in 1990. He is a member of the American Academy of Arts and Sciences, the National Academy of Engineering and the National Academy of Sciences. He has received honorary degrees from Link\"oping University in Sweden, the Polytechnic University of Madrid, Colby College, Trinity College, Dublin and Concordia University in Montreal. He has been declared a Distinguished Alumnus by the California Institute of Technology.

54. Chris McCraw

Network administrator for the Texas Institute of Computational & Applied Mathematics; contract work-Internet/heterogeneous networking/Internet + Unix focus.

55. Steven W. McDougall
Education:
BSEE, Princeton University 1982; MSE, Stanford University 1983
Employment:
1983-1987 Electrical Engineer, General Computer Co. Waltham MA; 1987-1990 Electrical Engineer, OsteoTechnology, Inc Framingham MA; 1990-1992 Software Engineer, Varityper, Billerica MA; 1992-1993 Software Engineer, Sepracor, Marlbourough MA; 1993-1995 Software Engineer, Genome Therapeutics,Waltham MA

56. Geoff Mendal
Vice President of Engineering
Tri-Pacific Consulting Corporation

Technical lead of a large-scale porting effort for Sun Microsystems, Inc.: the SunSoft Workshop (C, C++, ProWorks, and TeamWare tools) is being ported from SunOS 5.x to HP-UX 9.0x. Co-author of 2 research/reference texts on the programming language Ada (Exploring Ada, volumes 1 and 2).

57. Emilio Millan

Software engineer for Central Data Corporation of Champaign, Illinois.

58. Clark Morgan

Clark O. Morgan is an independent consultant and full-time computer software engineer, currently working on a contract basis with Intel Corp. I have been working full-time in the computer industry since 1984. My particular areas of interest are language recognition and the development of embedded, real-time hardware/software interfaces. My accomplishments include the development of an instrumentation control language for a digital oscilloscope (at Tektronix, Inc.), the development and maintenance of a Motif-compatible e-mail browser (as an employee of Mentor Graphics, Corp.), and the development of a paperless "run card" for Intel's premier wafer fabrication research facility in Aloha, Oregon.

59. Neil Vincent Murray
Associate Professor, Department of Computer Science, SUNY at Albany, Albany, New York 12222, 1987-present.
Assistant Professor, Department of Computer Science, SUNY at Albany, Albany, New York 12222, 1982-1987.
Assistant Professor, Computer Science Department, LeMoyne College, Syracuse, New York 13214, 1979-1982.
Cornell University, Ithaca, New York 1966-1970, B.S. Engineering Physics, June 1970
Syracuse University, Syracuse, New York 1971-1979
M.S. Computer and Information Science, January 1974,
Ph.D. Computer and Information Science, August 1979

60. Russell Nelson

Russell Nelson has been programming professionally since 1975. He has written at least one of nearly every type of program, including compilers, interpreters, macro-based text editors, and other types of programs related to this brief. His work has been recognized by PC Magazine for Technical Excellence in 1990. Somewhere between a hundred thousand and a million people are using his software.

61. Jeffrey F. Painter
B.S., Mathematics, Harvey Mudd College, 1973
M.S., Computer Science, University of Wisconsin-Madison, 1977

Ph.D., Applied Mathematics University of Wisconsin-Madison, 1979
Mathematician, Lawrence Livermore National Laboratory (University of California), 1979-present. I write computer codes for scientific simulations. I also developed a user interface and input language for a program that partly automates the development of scientific simulation codes. Articles I wrote are in journals and books on mathematics, engineering, and computer science.

62. Kent M. Pitman
Principal Technical Consultant, Harlequin Inc. [*]
Project Editor [*] and International Representative [*] for subcommittee X3J13 (ANSI programming language Common Lisp, X3.226). US Representative [*] to and Project Editor [*] for subcommittee ISO/IEC JTC 1/SC 22/WG 16 (work in progress toward design of ISO programming language ISLISP).
Co-Author of "Revised Report on the Algorithmic Language Scheme", MIT AI Memo 848b, Cambridge, MA, Jul-Sep 1991.

* = Title indicated shows affiliation but the signature offered here is not offered as official act of representation under that title. The title is offered here for purpose of identification and establishment of credentials only.

63. Steven Pothier

Mr. Pothier is a Senior Computer Scientist for the SAIC Image Technology and Systems Division. He works on state-of-the-art software systems incorporating object-oriented languages, advanced decision techniques, reconfigurable user interfaces, and genetic programs. Mr. Pothier is an expert-level C, and LISP programmer and has considerable experience in many other high and low-level languages many of which are high-level (4GL) languages. He has several years experience programming SIMD and MIMD parallel processors, including experience with data-parallel languages for the Connection Machine Model 2 (*Lisp, C/Paris, C*).

64. John Ramsdell

John Ramsdell is a Lead Scientist at The MITRE Corporation, Bedford MA. He has been employed there for 12 years. In 1982, he received his Ph.D. in Applied Mathematics, concentrating in numerical and discrete methods, from Harvard University. In 1976, he received his B.S., in Engineering Physics, from Cornell University.

At MITRE, John Ramsdell specializes in parallel and functional programming, high performance computation and communication, and formal verification. His activities have contributed to the construction of a verified processor which executes functional programs, and the construction of a formally verified implementation of a programming language (Scheme). He is involved in an effort to replace special purpose hardware for signal processing by commercial massively parallel processors.

65. Lyle Ramshaw

After getting my PhD under Prof. Donald E. Knuth at Stanford in 1979, I have been doing computer science research, first for Xerox Corporation and now for Digital Equipment Corporation. I pioneered "blossoming", a new approach to the mathematical theory underlying the "spline" curves and surfaces that are used in computer-aided geometric design.

66. Travis Raybold

I am a computer programmer, dealing mainly in cgi scripting for the world wide web.

67. Mark Riordan

Mark Riordan is the original author of the popular freeware Usenet newsreader WinVN, and of a free encryption program named RIPEM.

68. Arnold Robbins

I am the maintainer of GNU AWK, a freely available implementation of the AWK computer programming language. My involvement with this program has greatly improved my programming and writing skills, and has allowed me to improve the lot of hundreds of thousands of people who use GNU AWK daily for their work.

69. Michael C. Robert
Project Leader

I have been programming user interfaces for 11 years and I am currently working for a consulting company that designs and builds mass-market applications for Fortune 500 companies. To be an effective programmer, I have had to also become quite familiar with users and their expectations and needs.

70. Erik Rosenthal

I am a professor in the department of mathematics at the University of New Haven; my research interests are in the area of automated deduction. My Ph.D. is from Univ. of Calif. at Berkeley.

71. Roger Schlafly

Independent software developer. Ph.D. in Mathematics, Univ. of California (Berkeley). Worked in the software industry for ten years.

72. Karl F. Sierka
Boulder, Colorado

I am currently working for an ISP, doing usenet news admin. I have been doing computer work since the early eighties. I have written several large programs in several languages, each requiring extensive interactions with users.

73. Rafael Sorkin

He has degrees in physics from Harvard (undergrad) and Caltech (grad). Prof. of Physics at Syracuse University, on editorial board of International J. of Theoretical Physics. One of first people to use computers for numerical General Relativity (in around 1973).

74. Todd Squires

Graduate of University of Illinois at Urbana, 1987, B.S. Computer Science. Working in the computer industry for 15 years. Major areas of concentration: Video games, development tools.

75. Gerald Jay Sussman

Gerald Jay Sussman is the Matsushita Professor of Electrical Engineering at the Massachusetts Institute of Technology. He received the S.B. and the Ph.D. degrees in mathematics from the Massachusetts Institute of Technology in 1968 and 1973, respectively. He has been involved in artificial intelligence esearch at M.I.T. since 1964. He has also worked in computer languages and in computer architecture and in VLSI design.

Sussman was the principal designer of the Digital Orrery, a machine designed to do high-precision integrations for orbital-mechanics experiments. Using the Digital Orrery, Sussman and Jack Wisdom, discovered numerical evidence for chaotic motions in the solar system. The Digital Orrery is now retired at the Smithsonian Institution in Washington DC.

Sussman is a coauthor (with Hal Abelson and Julie Sussman) of, "Structure and Interpretation of Computer Programs," the introductory computer science textbook used at M.I.T. As a result of his contributions to computer-science education, Sussman received the ACM's Karl Karlstrom Outstanding Educator Award in 1990, and the Amar G. Bose award for teaching in 1991.

76. Dr. Christopher Vickery

Dr. Christopher Vickery is professor of computer science at Queens College of CUNY. He is the author of the volume, "Real-Time and Systems Programming for PCs" (McGraw-Hill, 1993), and is listed in the current edition of Marquis' Who's Who in the East. Dr. Vickery has developed a number of software development tools for microprogrammable computers and real-time systems, which have been distributed freely to coworkers in the industry.

77. Philip Wadler

Philip Wadler is a Professor of Computing Science at the University of Glasgow. He serves as editor-in-chief of the Journal of Functional Programming, published by Cambridge University Press; and wrote with Richard Bird 'Introduction to Functional Programming', which has been translated into Dutch, German, and Japanese. He helped design Haskell, the most widely used lazy functional programming language; and currently consults with Ericsson Telecommunications on the design of a type system for Erlang, a concurrent functional language used in phone switches. He has been an invited speaker at conferences in Gdansk, London, New Haven, Portland, Santa Fe, and Sydney; and was an ACM distinguished lecturer 1989-1993. He previously held a research post at Oxford University, and the title of his doctoral dissertation from Carnegie-Mellon was 'Listlessness is better than laziness'.

78. John Wardale

BSECE 1983, MSCS 1985 [UW-Madison]
Jobs: Engineering Computing Lab
Computer Aided Engineering Center
Astronautics Technology Center [unix-OS group member for the ZS-1 mini-super computer group]
Persoft Inc. [R&D team member—application programer]

79. Joel Welling

Joel Welling holds a PhD in gravitational physics. For almost a decade he has worked at the Pittsburgh Supercomputing Center, where he writes software for scientific computer graphics in support of the research of academic scientists in a number of locations across the US. He has written several computer graphics packages (including some supporting the CGM standard graphics metafile, the P3D system for three dimensional graphics, and the VFleet volume rendering package) which are available free via the Internet.

80. Ben White

Ben White is a computer scientist, with an M.S. degree from San Jose State University. He has 11 years of experience developing both computer software and hardware, including development of a visual programming environment called "Flowtran". He is currently employed with a maker of semiconductor manufacturing equipment. He has been a member of the League for Programming Freedom for four years.

81. Jan Zytkow

Jan Zytkow received his Ph.D. from the University of Warsaw in Philosophy of Science in 1972, and his habilitation in 1979. Until 1982 he worked at the University of Warsaw, researching and teaching logic, methodology of science, and philosophy. From 1982, for two years he was a Visiting Professor at Carnegie-Mellon University in Pittsburgh, where he worked on machine discovery, collaborating with Herbert Simon and Pat Langley. In 1984 he joined the Wichita State University where he has been a Professor of Computer Science and in addition to his research on discovery, he worked on autonomous decision making by automated pilot, on learning mechanisms for automated pilot, and on computer vision. He founded a machine discovery laboratory in which automated computer systems together with robotic hardware perform experiments and build theories in chemistry, mechanics and the like. Other research activities in the laboratory include discovery of hidden structure, discovery in databases, and discovery in geometry.